PROGRAMMING

SDK Guide for DX4100/DX4100/ DX4500/DX4600

Document Revision V1.0.5

C4616M

Overview *4*

Data Types *6*

SDK Functions *14*

Messages *24*

Pelco's API & SDK Document and Software License Agreement 26
Terms and Conditions 26
Questions 28

Contact Information 30

Overview

A fully featured and fully affordable entry-level digital video recorder (DVR), the DX4100/DX4500/DX4600 Series is the next generation in DVRs. Equipped with an embedded operating system, it offers camera capacity, features, and functionality exceeding the other DVRs. The DX4100/DX4500/DX4600 is designed for the entry-level market that requires 8 or 16 camera inputs; greater internal hard drive storage capacity; fast frame rate recording; and powerful playback, search, and export capability. The DX4100/DX4500/DX4600 not only replaces the traditional VCR and multiplexer combination, but it also offers the benefits derived from the latest in digital video processing.

The object of this documentation is to provide quick reference to software developers who need to integrate their software with the DX4500_DX4600 Series API using the provided SDK.

Data Types

The following are datatypes relevant to the DX4500/4600 API.

ST_SERVER_CONNECT_INFO

This data type represent network configuration data for connecting to a DX4100/DX4500/DX4600.

```
typedef struct _tagServerConnectInfo
{
     CHAR szServerAddr[SIZE_SERVER_ADDRESS];
     WORD wServerPort;
     CHAR szUserID[SIZE_USER_ID];
     CHAR szUserPasswd[SIZE_USER_PASSWORD];
     LPVOID lpfnConectCallback;
     LPVOID lpfnDisconnectCallback;
      unsigned long notifyHandle;
}ST_SERVER_CONNECT_INFO, FAR *LPST_SERVER_CONNECT_INFO;
```

szServerAddr: The DX4100/DX4500/DX4600's IP address.

wServerPort: The DX4100/DX4500/DX4600's port.

szUserID: The username to login to the DX4100/DX4500/DX4600.

szUserPasswd: The user's password to login to the DX4100/DX4500/DX4600.

lpfnConectCallback: The function address of returning a result of callback for connecting.

lpfnDisconnectCallback: The function address of returning a result of callback for disconnecting.

notifyHandle: Window handle that receives event when notifications are triggered.

ST_CONNECT_INFO

This data structure holds the result for both connecting to and logins into the DX4500/4600.

```
typedef struct _tagConnectInfo
{
     INT  nConnectResult;
     INT  nLoginResult;
}ST_CONNECT_INFO, FAR *LPST_CONNECT_INFO;
```

nConnectResult: An integer value that represents the connection request result. The following are possible valid values:

- **0 for NET_CONNECTED)**: A successful connection to the DX4500/4600 was established.
- **1 for NET_CONNECT_FAIL**: A connection to the DX4500/4600 was unable to be made.

nLoginResult: An integer value that represents the login request result. The following are possible valid values:

- **2 for NET_LOGIN_SUCCESS**: A successful login to the DX4500/4600 was made.
- **3 for NET_LOGIN_FAIL**: Unable to make a successful login to the DX4500/4600.

ST_SYSTEM_INFO

This data type contains information pertaining to the DX4500/4600.

```
typedef struct _tagSystemInfo
{
     WORD wTimeZone;
     WORD wCamera;
     WORD wAudio;
     WORD wVideoStandard;
} ST_SYSTEM_INFO,FAR* LPST_SYSTEM_INFO;
```

wTimeZone: The DX4500/4600's time zone as denoted by an integer representing the time zone's different from UTC.

wCamera: The number of cameras associated with the DX4500/4600.

wAudio: The number of audio sources associated with the DX4500/4600.

wVideoStandard: The tv video format being used by the DX4500/4600. Possible values are

- **0 for NTSC**
- **1 for PAL**

ST_CAMERA_INFO

This data type contains information relevant to a camera associated with a DX4500/4600.

```
typedef struct _tagCameraInfo
{
     LPVOIDlpfnCameraCallback;
}ST_CAMERA_INFO,FAR * LPST_CAMERA_INFO;
```

lpfnCameraInfoCallback: The address of the returning a result of a callback.

ST_EACH_CAMERA_INFO

This data type contains information relevant to a camera associated with a DX4500/4600.

```
typedef struct _tagEachCameraInfo
{
     CHAR szCamName[32];
     WORD wWidth;
     WORD wHeight;
     BYTE byFramerate;
     BYTE bySchedule;
     BYTE byQuality;
     BYTE byAudio;
     BYTE byVideoStandard;
     BYTE byCovert;
     BYTE byPTZ;
     BYTE byVideoLoss;
     INT  nAlarmIn;
     INT  nAlarmOut;
}ST_EACH_CAMERA_INFO,FAR *LPST_EACH_CAMERA_INFO;
```

szCamName: The camera's name.

wWidth: The width of camera's video.

wHeight: The height of camera's video.

byFramerate: The speed of the camera's video.

bySchedule: The camera's video schedule, specifically when the camera will begin recording video. The following are valid values:

- 0 for No Recording
- 1 for Normal
- 2 for Motion
- 3 for Alarm Input
- 4 for Event
- 5 for Data

byQuality: The camera's video's image quality.

- 0 for Lowest
- 1 for Low
- 2 for Normal
- 3 for High
- 4 for Best

byAudio: The camera's audio mapping.

- 0 for No Use
- 1 - 8 for a specific audio channel

byVideoStandard: The tv video standard/format being used.

- 0 for NTSC
- 1 for PAL

byCovert: video coversion??

- 0 for disabled
- 1 for enabled

byPTZ: pan, tilt, zoom

- 0 for disabled
- 1 for enabled

byVideoLoss: video loss detection

- 0 for disabled
- 1 for enabled

nAlarmIn: The number of alarm inputs.

nAlarmOut: The number of alarm outputs.

ST_HDD_INFO

This represents the DX4500/4600's hard drive status.

```
typedef struct _tagHDDInfo
{
    UINT bHDDConOverwriting;
    UINT nHDDConTotal;
    UINT nHDDConCurrUsage;
    UINT bHDDEvtOverwriting;
    UINT nHDDEvtTotal;
    UINTnHDDEvtCurrUsage;
}ST_HDD_INFO,FAR *LPST_HDD_INFO;
```

bHDDConOverwriting: Determines whether the HDD is continuously overwriting older data. Possible valid values are:

- 0 for no continuous overwriting
- 1 for continuous overwriting

nHDDConTotal: The size of continuous video space on the HDD.

nHDDConCurrUsage: The amount of continuous video space being used on the HDD.

bHDDEvtOverwriting: Determines whether the HDD overwrites event related video.

nHDDEvtTotal: The size of event related video space on the HDD.

nHDDEvtCurrUsage: The amount of event related video space being used on the HDD.

ST_REQ_VIDEO

```
typedef struct _tagReqLiveVideo
{
     int nchannel;
     LPVOID lpfnLiveCallback;
     LPVOID lpfnStopLiveCallback;
}ST_REQ_LIVE_VIDEO, FAR *LPST_REQ_LIVE_VIDEO;
```

nchannel: The 16 bit channel mask

lpfnLiveCallback: The function address of returning the result of a live callback.

lpfnStopLiveCallback: The function address of returning a result of stopping a live callback.

ST_DATA_FORMAT

```
typedef struct _tagDataFormat
{
     Time_t tm;
     int    nWidth;
     int    nHeight;
     BYTE   origintype;
     ORIGIN origin;
     BYTE   codectype;
     BYTE   savetype;
     DWORD  size;
     DWORD  dwInterval;
     LPBYTE realdata;
} ST_DATA_FORMAT, FAR *LPST_DATA_FORMAT;
```

tm: The video image's creation time.

nWidth: The width of video image.

nHeight: The height of video image

origintype: The type of data type ?????

- _D_VIDEO for video data
- _D_AUDIO for audio data
- _D_TEXT for text data
- _D_SENSOR for sensor
- _D_RECT for motion detection area
- _D_UNKNOWN for reserved

origin: An Origin data type.

codectype: The type of codec. e.g. S50X

savetype: The type of recording. The following are possible valid values:

- NO_RECORDING
- _R_NORMAL
- _R_MOTION
- _R_ALARM
- _R_DATA

size: The size of the data.

dwInterval: The interval of every other frame.

realdata: A pointer to 'real data'.

ORIGIN

Holds the amount of cameras and sensors.

```
typedef union _tagORIGIN
{
    BYTE cno;
    BYTE sno;
} ORIGIN;
```

cno: The number of cameras.

sno: The number of sensors.

ST_DATA_INFO_OF_DATE

```
typedef struct _tagDataInfoOfDate
{
    BOOL bHasData[32];
}ST_DATA_INFO_OF_DATE, FAR *LPST_DATA_INFO_OF_DATE;
```

bHasData: True if the DX4500/4600 has a video recording from first or last day of a requested month, false otherwise.

ST_DVR_TIMETABLE

```
typedef struct _tagDVRTimetable
{
    char tagTTMinData[16*25*60];
}ST_DVR_TIMETABLE, FAR *LPST_DVR_TIMETABLE;
```

tagTTMinData: This is timetable data. e.g. 16channels * 25hours * 60minutes

ST_PLAY_BACK

```
typedef struct _tagPlayback
{
    int nChannel;
    SYSTEMTIME startTime;
    int nPlaySpeed;
    BOOL      bAudioUse;
    BOOL      bMultiCh;
    LPVOID lpfnPlaybackCallback;
```

```
}ST_PLAY_BACK, FAR *LPST_PLAY_BACK
```

nChannel: The playback channel being used.

- -1 is all channels
- 0 ~ 15

startTime: The video's playback start time.

nPlaySpeed: The video's playback speed. The following are valid values:

- -8
- -4
- -2
- 1
- 2
- 8

bAudioUse: True to enable audio in playback, false otherwise.

BMultiCh: True to have all channels in playback, false otherwise.

lpfnPlaybackCallback: The function address of returning a result of a playback related callback.

ST_SMART_SEARCH_OPTION

This represents a search query.

```
typedef struct _tagSmartSearchOption
{
    SYSTEMTIME tmStart;
    SYSTEMTIME tmEnd;
    int nChannel;
    int nSensitivity;
    unsigned char btMotionArea[SIZE_SMART_SEARCH_AREA];
    LPVOID lpfnSmartSearchCallback
} ST_SMART_SEARCH_OPTION, FAR *LPST_SMART_SEARCH_OPTION;
```

tmStart: The start time of a recorded video.

tmEnd: The end time of a recorded video.

nChannel: The channel number to filter video results. The following are valid possible values: 0-15.

nSensitivity: The sensitivity to filter video results.

btMotionArea: The motion area to filter video results.

lpfnSmartSearchCallback: The function address of returning a result of smart search callback.

ST_PTZ_CTRL

```
typedef strcut _tagPTZCtrl
{
    int  nChannel;
    CHAR PTZCmd[3];
    int  nPreset;
} ST_PTZ_CTRL, FAR FAR *LPST_PTZ_CTRL;
```

nChannel: The channel number of the PTZ control.

PTZCmd: The PTZ command.

nPreset: The Preset.

ST_EVENT_LOG_LIST

Represents an event log item.

```
typedef struct _tagEventLogList
{
    SYSTEMTIME tmEvent;
    int  nChannel;
    int  nEventType;
} ST_EVENT_LOG_LIST, FAR *LPST_EVENT_LOG_LIST
```

tmEvent: The time of the event.

nChannel: The channel number where the event occurred. The following are valid possible values:

- all channels = -1
- 0 ~ 15

nEventType: The type of event. The following are valid possible values:

- 1 for all (motion & sensor)
- 2 for motion
- 3 for sensor(alarm in)

ST_SYSTEM_LOG_ITEM

```
typedef struct _tagSystemLogItem
{
    SYSTEMTIME tmLog;
    int nLogType;
} ST_SYSTEM_LOG_ITEM, FAR *LP ST_SYSTEM_LOG_ITEM;
```

tmLog: The time of the log entry.

nLogType: The type of log entry. The following are possible valid values:

- EVENT_FILTER_ALL for 0x00
- EVENT_TYPE_ALARM_IN for 0x01
- EVENT_TYPE_MOTION for 0x02
- EVENT_TYPE_ATMPOS for 0x04

ST_BOOKMARK_ITEM

```
typedef strcut _tagBookmarkItem
{
    SYSTEMTIME tmBookmark;
    unsigned int          nChMap;
} ST_BOOKMARK_ITEM, FAR *LPST_BOOKMARK_ITEM;
```

tmBookmark: The time of the bookmark.

nChMap: The bookmark's channel mask.

SDK Functions

Please refer to the previous 'Data Types' section for more details on any data types mentioned for SDK functions.

DVR_INITIALIZE

This initializes the DX4500/4600.

```
VOID DVR_Initialize();
```

DVR_CLOSE

This turns off the DX4500/4600??

```
VOID DVR_Close();
```

DVR_CONNECT

Connects to the DX4500/4600 with guest permissions and returns the connected server's handle.

```
HSERVER DVR_Connect(LPST_SERVER_CONNECT_INFO lpServerConnectInfo, BYTE byStatus;);
```

lpServerConnectInfo: (in) Information of server to connect.

byStatus: (out) The connection status to the DX4500/4600. The following are valid possible values:

- SOCKET_DISCONNECTED
- SOCKET_DISCONNECTING
- SOCKET_CONNECTING
- SOCKET_CONNECTED

DVR_DISCONNECT

Disconnects existing connection to the DX4500/4600 and returns the connected server's handle.

```
VOID DVR_Disconnect(HSERVER hServer, int nCause);
```

hServer: (in) Connected server handle

nCause: (in) The reason for closing socket. The following are possible valid values:

- 0 for CCC_EXIT: Closing Socket by Exit Thread
- 1 for CCC_CONNECT: Closing Socket with Error in Connect
- 2 for CCC_SEND: Closing Socket with Error in Send
- 3 for CCC_RECEIVE: Closing Socket with Error in Receive
- 4 for CCC_MESSAGE: Closing Socket with Invalid Message Format
- 5 for CCC_MEMORY: Closing Socket with Insufficient Memory to Allocate
- 6 for CCC_REDIRECT: Closing Socket to Reconnect to Another Site

DVR_USERLOGIN

This connections to the DX4500/4600 using a given IP address, port, username, and password; and returns the handle to the connected server.

```
HSERVER DVR_UserLogin( LPST_SERVER_CONNECT_INFO lpServerInfo );
```

lpServerInfo: (in) Information needed to make a connection to the server.

DVR_ONCONNECTCALLBACK

A callback function that returns the result of a login.

```
VOID DVR_OnConnectCallback(HSERVER hServer, LPST_CONNECT_INFO lpConnectInfo);
```

hServer: (in) A connected server handle.

lpConnectInfo: (in) Information related to the login.

DVR_ONCONNECTRESETCALLBACK

A callback function that returns the result of a unexpected network disconnection.

```
VOID  DVR_OnConnectResetCallback(HSERVER hServer);
```

hServer: (in) A connected server handle.

LOGOUT

Disconnects from the DX4500/4600.

```
VOID DVR_Logout(HSERVER hServer, int nCause);
```

hServer: (in) A connected server handle.

nCause: (in) Contains a status of why a socket closed.

- 0 for CCC_EXIT: Socket closed due to thread exit.
- 1 for CCC_CONNECT: Socket closed due to an error in Connect.
- 2 for CCC_SEND: Socket closed due to an error in Send.
- 3 for CCC_RECEIVE: Socket closed due to an error in Receive.
- 4 for CCC_MESSAGE: Socket closed due to an invalid message format.
- 5 for CCC_MEMORY: Closing Socket with Insufficient Memory to Allocate
- 6 for CCC_REDIRECT: Closing Socket to Reconnect to Another Site

DVR_GETSYSTEMINFO

Returns the DX4500/4600 system info. If the function call is successful it will return a 1, otherwise it will return a 0.

```
INT DVR_GetSystemInfo(HSERVER hServer, LPST_SYSTEM_INFO lpstSystemInfo);
```

hServer: (in) A connected server handle.

lpstSystemInfo: (out) Information related to DX4500/4600 system. Please refer to the ST_SYSTEM_INFO in the 'Data Types' section for details.

DVR_ONCAMERACALLBACK

A callback function that is called automatically when a client request camera information from the DX4500/4600.

```
VOID DVR_OnCameraCallback(HSERVER hServer, LPST_EACH_CAMERA_INFO lpstEachCameraInfo);
```

hServer: (out) A connected server handle.

lpstEachCameraInfo: (out) Information regarding a specific camera. Please refer to ST_EACH_CAMERA_INFO in the 'Data Types' section for more details.

DVR_GETHDDINFO

Return the information regarding the DX4500/4600's hard drive. If the function call is successful, it will return a 1; otherwise it will return a 0.

```
INT DVR_GetHDDInfo(HSERVER hServer, LPST_HDD_INFO lpstHddInfo);
```

hServer: (in) A connected server handle.

lpstHddInfo: (out) Hard drive related information. Please refer to ST_HDD_INFO in the 'Data Types' section.

DVR_BEGINLIVEVIDEO

This requests live video streaming from the DX4500/4600.

***NOTE**: This function does not support audio.*

```
VOID DVR_BeginLiveVideo(HSERVER hServer, LPST_REQ_LIVE_VIDEO lpReqLiveVideo);
```

hServer: (in) A connected server handle.

lpLiveVideo: (in) Information related live video. Please refer to ST_REQ_LIVE_VIDEO in the 'Data Types' section for details.

DVR_ ONDATACALLBACK

A callback function this is called automatically when clients request live and audio data from the DX4500/4600.

```
VOID DVR_OnLiveCallback(HSERVER hServer, LPST_DATA_FORMAT lpDataFormat, LPBYTE lpRealData);
```

hServer: (out) A connected server handle.

lpDataFormat: (out) Please refer to LPST_DATA_FORMAT in the 'Data Types' section.

lpRealData: (out) A pointer to real video or audio data.

DVR_ONSTOPLIVECALLBACK

A callback function that is called automatically when a client gets the result of a stop live video request.

```
VOID DVR_OnStopLiveCallback(HSERVER hServer, BOOL bResult);
```

hServer: (out) A connected server handle.

bResult: (out) The result of a stop video request.

DVR_STOPLIVEVIDEO

This makes a request to stop live video stream from the DX4500/4600. The function call is successful if it returns a 1, otherwise it will return a 0.

```
INT DVR_StopLiveVideo(HSERVER hServer);
```

hServer: (in) A connected server handle.

DVR_GETRECORDEDDAY

Returns information regarding a specific day of recording within a particular month. If successful this will return a 1, otherwise it will return 0.

```
INT DVR_GetRecordedDay(HSERVER hServer, SYSTEMTIME *pRequestTime, LPST_DATA_INFO_OF_DATE lpstDataofDate);
```

hServer: (in) A connected server handle.

pRequestTime: (in) Contains the requested time, year, and month.

lpstDataofDate: (in) Information related to date of video recording. Please refer to ST_DATA_INFO_OF_DATE in the 'Data Types' section.

DVR_GETTIMETABLE

This retrieves the DX4500/4600's time table. If successful the function will return a 1, otherwise it will return a 0.

```
INT DVR_GetTimetable(HSERVER hServer, SYSTEMTIME *pRequestTime, LPST_DVR_TIMETABLE lpstTimeTable);
```

hServer: (in) A connected server handle.

pRequestTime: (in) The requested time.

lpstTimeTable: (out) Information of timetable. Please refer to ST_DVR_TIMETABLE in the 'Data Types' section.

DVR_SETPLAYBACKOPTION

Allows a client to set options regarding video playback. If successful the function will return a 1, otherwise it will return a 0.

NOTE*: If a user wants to play a video from a new server, the user must call this function to remove current playback information.Then the user must call the DVR_SetPlaybackOption() function.*

```
INT DVR_RemovePlaybackOption(HSERVER hServer);
```

hServer: (in) A connected server handle.

DVR_PLAYBACKCONTROL

This sends a video playback command to the DX4500/4600. If successful the function will return a 1, otherwise it will return a 0.

NOTE*: For this function to be successful, the DVR_SetSearchPlaybackOption function must be called in advance.*

```
VOID S5PlaybackControl(HSERVER hServer, int nCtrlCmd);
```

hServer: (in) A connected server handle.

nCtrlCmd: (in) This playback control value. The following are valid possible values:

- DVR_PLAYBACK _FPLAY: Forward Playback.
- DVR_PLAYBACK _BPLAY: Reverse Playback.
- DVR_PLAYBACK _NEXTFRAME: Next frame.
- DVR_PLAYBACK_PREVFRAME: Prev frame.
- DVR_PLAYBACK_FIRSTFRAME: First frame.
- DVR_PLAYBACK_LASTFRAME: Last frame.
- DVR_PLAYBACK_PAUSE: Playback pause
- DVR_PLAYBACK_STOP: Playback stop

DVR_REQUESTSMARTSEARCH

This returns the result of the smart search request on enabled cameras that connect to the server remotely. If successful the function will return a 1, otherwise it will return a 0.

```
INT DVR_RequestSmartSearch(HSERVER hServer, LPST_SMART_SEARCH_OPTION lpSmartSearchOption);
```

hServer: (in) A connected server handle.

lpSmartSearchOption: (in) Hold options for smart search. Please refer to ST_SMART_SEARCH_OPTION in the 'Data Types' section for details.

NOTE*: Please keep in mind the following:*

*#define SIZE_SMART_SEARCH_AREA SIZE_SMART_SEARCH_AREA_X * SIZE_SMART_SEARCH_AREA_Y*

#define SIZE_SMART_SEARCH_AREA_X 1

#define SIZE_SMART_SEARCH_AREA_Y 9

DVR_ONSMARTSEARCHCALLBACK

A callback function that retrieves the smart search result from the DX4500/4600.

```
VOID DVR_OnSmartSearchCallback(HSERVER hServer, LPST_SMART_SEARCH_INFO lpstSmartSearchInfo);
```

hServer: (out) A connected server handle.

lpstSmartSearchInfo: (out) Please refer to LPST_SMART_SEARCH_INFO in the 'Data Types' section.

DVR_ABORTSMARTSEARCH

This function makes a request to stop an in progress smart search.

```
VOID DVR_AbortSmartSearch( void );
```

DVR_PTZCONTROL

This function sends pan, tilt, and zoom requests to a specific camera associated with the DX4500/4600. If successful the function will return a 1, otherwise it will return a 0.

```
INT DVR_PTZControl(HSERVER hServer, LPST_PTZ_CTRL lpPTZCtrl);
```

hServer: (in) Connected server handle

lpPTZCtrl: (in) Contains the PTZ command. The following are valid possible values:

- DVR_PTZ_UP: Tilt upward.
- DVR_PTZ_DOWN: Tilt downward.
- DVR_PTZ_LEFT: Pan left.
- DVR_PTZ_RIGHT: Pan right.
- DVR_PTZ_ZOOMIN: Operation of zoom (zoom in).
- DVR_PTZ_ZOOMOUT: Operation of zoom (zoom out).
- DVR_PTZ_FOCUSIN: Operation of focus (focus in).
- DVR_PTZ_FOCUSOUT: Operation of focus (focus out).
- DVR_PTZ_IRIS_PLUS: Operation of iris (iris plus).
- DVR_PTZ_IRIS_MINUS: Operation of iris (iris minus).
- DVR_PTZ_START: Start of PTZ operation.
- DVR_PTZ_END: End of PTZ operation.
- DVR_PTZ_PRESET_SET: Operation of preset set
- DVR_PTZ_PRESET_CLEAR: Operation of preset clear.
- DVR_PTZ_PRESET_MOVE: Operation of preset move.
- DVR_PTZ_PRESET_TOUR: Operation of preset tour.
- DVR_PTZ_STOP: Operation of PTZ stop.
- DVR_PTZ_SETUP_START: Operation of PTZ menu start.
- DVR_PTZ_SETUP_END: Operation of PTZ menu end.
- DVR_PTZ_SETUP_ENTER: Operation of PTZ menu enter.

DVR_GETFIRSTEVENTLOGLIST

This function retrieves the 1st 30 event log entry results from the DX4500/4600. If successful the function will return a 1, otherwise it will return a 0.

```
INT DVR_GetFirstEventLogList(HSERVER hServer, SYSTEMTIME *tmEventDate, int nChannel, int nFilter,
     LPST_EVENT_LOG_LIST stEventLogList);
```

hServer: (in) A connected server handle.

tmEventDate: (in) The date used to filter event entry results.

nChannel: (in) The channel number used to filter event entry results.

nFilter: (in) Event type filter used to limit event entry results. Please refer to DVR_EVENT_FILTER in the 'Data Types' section.

stEventLogList: (out) The event log list to return.

***NOTE**: 'COUNT_EVENT_LOG' is the number of event log entries to return, and is fixed at 30.*

DVR_GETNEXTEVENTLOGLIST

This function retrieves the next 30 event log entry results from the DX4500/4600. If successful the function will return a 1, otherwise it will return a 0.

***NOTE**: The DVR_GetFirstEventLogList() function must be called in advance before this function.*

```
INT DVR_GetNextEventLogList(HSERVER hServer, LPST_EVENT_LOG_LIST stEventLogList);
```

hServer: (in) A connected server handle.

stEventLogList: (out) The event log entries list to return.

DVR_GETPREVEVENTLOGLIST

This function retrieves the previous 30 event log entry results from the DX4500/4600. If successful the function will return a 1, otherwise it will return a 0.

***NOTE**: The DVR_GetFirstEventLogList() function must be called in advance before this function.*

```
INT DVR_GetPreviousEventLogList(HSERVER hServer, LPST_EVENT_LOG_LIST stEventLogList);
```

hServer: (in) A connected server handle.

stEventLogList: (out) The event log entries list to return.

DVR_GETSYSTEMLOGLIST

This function retrieves the 1st 30 system log entry results from the DX4500/4600. If successful the function will return a 1, otherwise it will return a 0.

```
INT DVR_GetSystemLogList(HSERVER hServer, SYSTEMTIME *tmLog, LPST_SYSTEM_LOG_ITEM lpSystemLogList);
```

hServer: (in) A connected server handle.

tmLog: (in) The date used to filter the results.

lpSystemLogList: (out) The list of system log entries to return. The following are valid possible log entries:

- LOG_POWERON: Power On
- LOG_POWEROFF: Power Off
- LOG_OADMIN_IN: Admin Login (Local)
- LOG_OPOWER_IN: Power user Login (Local)
- LOG_ONORMAL_IN: Normal user Login (Local)

- LOG_CADMIN_IN: Admin Login (Remote)
- LOG_CPOWER_IN: Power user Login (Remote)
- LOG_CNORMAL_IN: Normal user Login (Remote)
- LOG_OADMIN_OUT: Admin Logout (Local)
- LOG_OPOWER_OUT: Power user Logout (Local)
- LOG_ONORMAL_OUT: Normal user Logout (Local)
- LOG_CADMIN_OUT: Admin Logout (Remote)
- LOG_CPOWER_OUT: Power user Logout (Remote)
- LOG_CNORMAL_OUT: Normal user Logout (Remote)
- LOG_CONFIG_CHANGED: Configuration changed
- LOG_CONFIG_IMPORT: Configuration imported
- LOG_CONFIG_FACTORY: Factory Default Set
- LOG_POWER_RECOVERY: Power Recovery
- LOG_BACKUP_START: Backup Started
- LOG_BACKUP_STOP: Backup Finished
- LOG_BACKUP_FAIL: Backup Failed
- LOG_EXPORT_START: Export Started
- LOG_EXPORT_STOP: Export Finished
- LOG_EXPORT_FAIL: Export Failed
- LOG_SW_UPDATED:SW Updated
- LOG_POWERFAN_FAIL: Poswer Fan Failure
- LOG_SYSTEMFAN_FAIL: System Fan Failure
- LOG_TEMPERATURE_FAIL: Temperature Failure
- LOG_BADHDD_DETECTED_PM: HDD Damaged (PM)
- LOG_BADHDD_DETECTED_PS: HDD Damaged (PS)
- LOG_BADHDD_DETECTED_SM: HDD Damaged (SM)
- LOG_BADHDD_DETECTED_SS: HDD Damaged (SS)
- LOG_HDD_ADDED_PM: HDD Added (PM)
- LOG_HDD_ADDED_PS: HDD Added (PS)
- LOG_HDD_ADDED_SM: HDD Added (SM)
- LOG_HDD_ADDED_SS: HDD Added (SS)
- LOG_HDD_REMOVED_PM: HDD Deleted (PM)
- LOG_HDD_REMOVED_PS: HDD Deleted (PS)
- LOG_HDD_REMOVED_SM: HDD Deleted (SM)
- LOG_HDD_REMOVED_SS: HDD Deleted (SS)
- LOG_HDD_FORMATTED_PM: HDD Fotmatted (PM)
- LOG_HDD_FORMATTED_PS: HDD Fotmatted (PS)
- LOG_HDD_FORMATTED_SM: HDD Fotmatted (SM)
- LOG_HDD_FORMATTED_SS: HDD Fotmatted (SS)
- LOG_HDD_REPLACED_PM: HDD Changed (PM)
- LOG_HDD_REPLACED_PS: HDD Changed (PS)
- LOG_HDD_REPLACED_SM: HDD Changed (SM)
- LOG_HDD_REPLACED_SS: HDD Changed (SS)

***NOTE**: 'COUNT_SYSTEM_LOG' is the number of system log entries to return, and is fixed at 30.*

DVR_GETNEXTSYSTEMLOGLIST

This function retrieves the next 30 system log entry results from the DX4500/4600. If successful the function will return a 1, otherwise it will return a 0.

***NOTE**: The DVR_GetFirstSystemLogList() function must be called in advance, before this function can be called.*

```
INT DVR_GetNextSystemLogList(HSERVER hServer, LPST_SYSTEM_LOG_ITEM lpSystemLogList);
```

hServer: (in) A connected server handle.

lpSystemLogList: (out) The system log list to return.

DVR_GETPREVIOUSSYSTEMLOGLIST

This function retrieves the previous 30 system log entry results from the DX4500/4600. If successful the function will return a 1, otherwise it will return a 0.

***NOTE**: The DVR_GetFirstSystemLogList() function must be called in advance, before this function can be called.*

```
INT DVR_GetNextSystemLogList(HSERVER hServer, LPST_SYSTEM_LOG_ITEM lpSystemLogList);
```

hServer: (in) A connected server handle.

lpSystemLogList: (out) The system log list to return.

DVR_GETBOOKMARKLIST

This function retrieves the bookmark list from the DX4500/4600. If successful the function will return a 1, otherwise it will return a 0.

***NOTE**: 'COUNT_BOOKMARK' is the number of bookmark entries and is fixed as 15.*

```
INT DVR_GetBookmarkList (HSERVER hServer, LPST_BOOKMARK_LIST lpBoomarkList,);
```

hServer: (in) Connected server handle

lpBoomarkList: (out) Bookmark list.

DVR_GETSOCKETLASTERROR

Retrieves the last-error code value regarding a socket. This function returns a specific error code when an error occurs, otherwise zero will be returned.

```
VOID DVR_GetSocketLastError(HSERVER hServer);
```

hServer: (in) A connected server handle.

APPE_LSOCKETCLOSED: A listening socket closed.

APPE_SOCKETNOTCREATED: A connection socket was not created.

APPE_INVALIDSOCKET: A connection socket was not created or closed already.

APPE_TIMEDOUT: A connection socket time out occurred while receiving data.

WSASYSNOTREADY: The underlying network subsystem is not ready for network communication.

WSAVERNOTSUPPORTED: The version of Windows Sockets support requested is not provided by this particular Windows Sockets implementation.

WSAEINPROGRESS: A blocking Windows Sockets operation is in progress.

WSAEPROCLIM: The limit on the number of tasks supported by the Windows Sockets implementation has been reached.

WSAEFAULT: The lpBuf argument is not in a valid part of the user address space.

WSANOTINITIALISED: A successful initialization (by WSAStartup() or AfxSocketInit()) must occur before using this API.

WSAENETDOWN: The Windows Sockets implementation detected that the network subsystem failed.

WSAEAFNOSUPPORT: The specified address family is not supported.

WSAEMFILE: No more file descriptors are available.

WSAENOBUFS: No buffer space is available. The socket cannot be connected.

WSAEPROTONOSUPPORT: The specified port is not supported.

WSAEPROTOTYPE: The specified port is the wrong type for this socket.

WSAESOCKTNOSUPPORT: The specified socket type is not supported in this address family.

WSAEADDRINUSE: The specified address is already in use.

WSAEADDRNOTAVAIL: The specified address is not available from the local machine.

WSAEINVAL: The socket has not been bound with Bind or The socket is already bound to an address.

WSAENOTSOCK: The descriptor is a file, not a socket

WSAEINTR: The (blocking) Windows Socket 1.1 call was canceled through WSACancelBlockingCall.

WSAEALREADY: A non-blocking connect call is in progress on the specified socket.

WSAECONNREFUSED: The attempt to connect was forcefully rejected.

WSAEISCONN: The socket is already connected.

WSAENETUNREACH: The network cannot be reached from this host at this time.

WSAETIMEDOUT: The attempt to connect timed out without establishing a connection.

WSAEWOULDBLOCK: The socket is marked as non-blocking and the connection cannot be completed immediately.

WSAEACCES: The requested address is a broadcast address, but the appropriate flag was not set.

WSAENETRESET: The connection must be reset because the Windows Sockets implementation dropped it.

WSAENOTCONN: The socket is not connected.

WSAEOPNOTSUPP: MSG_OOB was specified, but the socket is not of type SOCK_STREAM.

WSAESHUTDOWN: The socket has been shut down; it is not possible to call Send on a socket after Shut-Down has been invoked with How set to 1 or 2.

WSAEMSGSIZE: The socket is of type SOCK_DGRAM, and the datagram is larger than the maximum supported by the Windows Sockets implementation.

WSAEHOSTUNREACH: The remote host cannot be reached from this host at this time.

WSAECONNABORTED: The virtual circuit was aborted due to timeout or other failure.

WSAECONNRESET: The virtual circuit was reset by the remote side.

WSAEDESTADDRREQ: A destination address is required.

Messages

When making a request to the DVR_Connect() function, the Window handle is transferred to notifyHandle and allows the PostMessage function to post the WM_DVR_NOTIFY message. When WM_DVR_NOTIFY is posted, the WPARAM value of the window message procedure is the event triggered from the DVR.

The following are valid possible first parameter values for WM_DVR_NOTIFY:

0 for DVR_VIDEO_LOSS: This indicates that the second parameter is the mask of the video channel. If this value is 0, all channels is video loss channel.

1 for DVR_SCHEDULE_INFO: This indicates that the camera schedule has changed, and that the second accompanying parameter is an array type structure that contains camera schedule information.

2 for DVR_RECORD_STATE: Record state. When you request DVR_BeginLiveVideo, OnNotifiedByDVR() receive automatically each channel's record status. Second parameter is array type structure that channel's record state information is inserted

3 for DVR_SW_INFO: SW version info. Second parameter is character pointer type that connected DVR server version information is inserted

4 for DVR_ALARM_INPUT: Alarm Input. When input Alarm Second parameter is mask information of channel that occur alarm input.

5 for DVR_VIDEO_MOTION: Motion channel. When occurred Motion. Second parameter is mask information of channel that occur motion.

When calling DVR_Connect(), the Window handle is transferred to notifyHandle, letting WM_DVR_ERROR message perform by PostMessage. When WM_DVR_ERROR is post message, The LPARAM value of window message procedure is the event resulted from DVR.

The following are valid possible second parameter values for LPARAM:

1 for DVR_USER_ID_ERROR: User name is not registered.

2 for DVR_USER_PW_ERROR: Invalid password

3 for DVR_USER_USED_ERROR: The current User is being used.

4 for DVR_USER_ADMIN_ERROR: Another user is logged in as the Administrator.

5 for DVR_RSEARCH_OK: The DVR has returned search results.

6 for DVR_RSEARCH_ERROR: There were no results for the search request.

7 for DVR_EVENT_ERROR: There is no data for the event.

8 for DVR_LOG_ERROR: There is no data for the log.

9 for DVR_RSEARCHINDEX_ERROR: There is no data for the timetable.

10 for DVR_BOOKMARK_ERROR: There is no data for the bookmark.

11 for DVR_PTZ_ERROR: Pan, tilt, zoom funtions have failed.

Pelco's API & SDK Document and Software License Agreement

THIS AGREEMENT IS A LEGAL CONTRACT BETWEEN YOU ("YOU" OR "LICENSEE") AND PELCO ("PELCO" OR "LICENSOR") GOVERNING YOUR USE OF THE PELCO APPLICATION PROTOCOL INTERFACE DOCUMENT ("API & SDK DOCUMENT") AND THE PELCO APPLICATION PROTOCOL INTERFACE SOFTWARE ("API & SDK SOFTWARE") (COLLECTIVELY, THE "API & SDK"). YOUR USE OF EITHER THE API & SDK DOCUMENT OR THE API & SDK SOFTWARE CONFIRMS YOUR ACCEPTANCE OF THIS AGREEMENT, AND BY USING EITHER THE API & SDK DOCUMENT OR THE API & SDK SOFTWARE, YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, THAT YOU UNDERSTAND AND AGREE THAT YOU ARE BOUND BY ITS TERMS AND CONDITIONS, AND THAT YOU HAVE REGISTERED WITH AND BEEN CERTIFIED BY PELCO TO USE THE API & SDK. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT, YOU WILL NOT BE LICENSED TO USE THE API & SDK.

TERMS AND CONDITIONS

1. LICENSE GRANT.

PELCO hereby grants to LICENSEE a non-exclusive, royalty-free license (the "License") to use the API & SDK for the sole purpose of the development of computer application programs designed to function with PELCO's applicable line of products (the "Product") and the integration of the same.

2. TERM; TERMINATION.

This License is effective until terminated. PELCO reserves the right to automatically terminate this License, without notice to LICENSEE, if LICENSEE fails to comply with any terms of this License. Upon termination, LICENSEE shall immediately return to PELCO the API & SDK Document and the original disk containing the API & SDK Software, as well as all copies thereof, and all printed and written materials relating thereto, at LICENSEE's expense.

3. API & SDK OWNERSHIP

PELCO will retain all right, title, and interest in and to the API & SDK, and all copies of the API & SDK, regardless of the media or form on or in which the API & SDK or other copies may exist, including copies which are made in violation of the terms of this License. PELCO does not sell any property rights in the API & SDK, and this License is not a sale of the API & SDK or any copy thereof.

4. INTELLECTUAL PROPERTY RIGHTS.

The API & SDK includes proprietary information of PELCO and is protected by copyright, trademark and patent laws and international treaties. YOU agree that any copies API & SDK Document and the API & SDK Software shall contain the same copyright and other proprietary notices as appear on the original API & SDK Document and the API & SDK Software. YOUR use of the PELCO's copyright and other proprietary notices, as well as the PELCO logo, is conditioned upon YOUR obtaining the prior written approval of PELCO for all such notices, markings and uses. Except as specifically stated in this License, this License does not grant YOU any rights to any copyrights, trademarks, trade secrets, patents or any other intellectual property rights. YOU may be held legally liable for any copyright infringement, trademark infringement or patent infringement which is caused or encouraged by YOUR failure to abide by the terms of this License.

5. RESTRICTIONS ON USE.

Without the express written consent of PELCO, LICENSEE may not: (a) use the API & SDK for video surveillance equipment or any other equipment, products or software which are not intended by PELCO to be integrated with the Product; (b) transfer the API & SDK to multiple video surveillance equipment components, except as expressly provided in the API & SDK Document; or (c) copy, decompile, disassemble, duplicate, reverse engineer, adapt, modify, translate, or otherwise create derivative works based on the API & SDK , or any portion or copy thereof.

6. RESTRICTIONS ON ASSIGNMENT OR TRANSFER

LICENSEE shall not rent, lease, sell, sublicense, pledge or encumber the API & SDK , or any portion thereof, or any accompanying materials, to another party.

7. API & SDK UPDATES.

PELCO is under no obligation under this License to provide any patches, fixes, subsequent versions, upgrades, or support (collectively "updates") for the API & SDK . However, updated versions of the API & SDK may be created or issued by PELCO from time to time. At its sole option, PELCO may make such updates available to LICENSEE. The API & SDK is subject to change without notice to LICENSEE.

8. TROUBLE-SHOOTING.

For the purposes of trouble-shooting and PELCO customer service, YOU agree to provide PELCO with access to any Software Driver developed by YOU to which the API & SDK Software will interface to in the Pelco family of products. PELCO agrees not to disclose any confidential information about the Software Driver(s) to any third-party not involved with the trouble-shooting.

9. EXPORT CONTROL LAWS.

YOU will not, directly or indirectly, and will not allow others to, export the API & SDK or any technical data, software, technology plans and specifications relating to the API & SDK : (i) in violation of any restrictions, laws, rules or regulations of the U.S. or of any other applicable country, including, but not limited to, U.S. export control laws such as the Export Administration Regulations, the International Traffic and Arms Regulations and the Bureau of Census Regulations; (ii) to any country, entity or individual which or who is subject to U.S. export restrictions, or export restrictions of any other applicable country; or (iii) if YOU know, have reason to know or have a reasonable suspicion that the API & SDK or any technical data, software, technology plans and specifications relating to the API & SDK may be provided to a country, entity or individual which or who is subject to U.S. export restrictions or export restrictions of any other applicable country. You agree to promptly report to PELCO any violation of any international trade regulation, including, but not limited to, U.S. export controls and U.S. import regulations enforced by any agencies in the U.S. and abroad. YOU agree that YOU will take all steps necessary to assist PELCO in whatever way necessary to comply with relevant laws and to deal with any government inquiries, investigations, penalty actions, litigation, administrative actions, etc. arising out of questions about international trade regulation compliance, including, but not limited to, U.S. export controls and U.S. import regulations enforced by any agencies in the U.S. and abroad; however, there will be no limitation on the extent and nature of information YOU warrant and agree to provide.

10. DISCLAIMER OF WARRANTIES.

YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT USE OF THE API & SDK IS AT YOUR SOLE RISK AND THAT THE ENTIRE RISK AS TO SATISFACTORY QUALITY, PERFORMANCE, ACCURACY AND EFFORT IS WITH YOU. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE API & SDK ARE PROVIDED "AS IS", WITH ALL FAULTS AND WITHOUT WARRANTY OF ANY KIND, AND PELCO HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH RESPECT TO THE API & SDK, EITHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES AND/OR CONDITIONS OF MERCHANTABILITY, OF SATISFACTORY QUALITY, OF FITNESS FOR A PARTICULAR PURPOSE, OF ACCURACY, OF QUIET ENJOYMENT, AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS. PELCO DOES NOT WARRANT AGAINST INTERFERENCE WITH YOUR ENJOYMENT OF THE API & SDK, THAT THE DIRECTIONS CONTAINED IN THE INFORAMTION WILL MEET YOUR REQUIREMENTS, THAT THE USE OF THE API & SDK WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT DEFECTS IN THE API & SDK WILL BE CORRECTED. PELCO DISCLAIMS ANY WARRANTIES REGARDING THE SECURITY, RELIABILITY, TIMELINESS, AND PERFORMANCE OF THE API & SDK . PELCO DOES NOT WARRANT THE RESULTS THAT WILL BE OBTAINED BY YOUR USE OF THE API & SDK . NO ORAL OR WRITTEN ADVICE GIVEN BY PELCO OR A PELCO AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY. SHOULD THE API & SDK PROVE DEFECTIVE, YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. YOU UNDERSTAND AND AGREE THAT YOUR USE OF THE API & SDK IS AT YOUR OWN DISCRETION AND RISK AND THAT YOU WILL BE SOLELY RESPONSIBLE FOR ANY DAMAGES TO YOUR PELCO SURVEILLANCE VIDEO EQUIPMENT AND ANY NON-PELCO PRODUCTS INTEGRATED WITH PELCO SURVEILLANCE VIDEO EQUIPMENT OR LOSS OF DATA THAT RESULTS FROM SUCH INTEGRATION.

SOME STATES OR OTHER JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSIONS MAY NOT APPLY TO YOU. YOU MAY ALSO HAVE OTHER RIGHTS THAT VARY FROM STATE TO STATE AND JURISDICTION TO JURISDICTION.

11. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES SHALL PELCO BE LIABLE TO YOU OR ANYONE ELSE ON ACCOUNT OF YOUR USE OR MISUSE OF OR RELIANCE ON THE API & SDK OR ARISING FROM ANY CLAIM RELATING TO THIS LICENSE OR THE SUBJECT MATTER HEREOF. SUCH LIMITATION OF LIABILITY SHALL APPLY TO PREVENT RECOVERY OF DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, SPECIAL, EXEMPLARY, AND PUNITIVE DAMAGES WHETHER SUCH CLAIM IS BASED ON WARRANTY, CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE (EVEN IF PELCO HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES). SUCH LIMITATION OF LIABILITY SHALL APPLY WHETHER THE DAMAGES ARISE FROM USE, MISUSE OR RELIANCE ON THE API & SDK, FROM INABILITY TO USE THE API & SDK, OR FROM THE INTERRUPTION, SUSPENSION, OR TERMINATION OF THE RIGHT TO USE THE API & SDK (INCLUDING SUCH DAMAGES INCURRED BY THIRD PARTIES). THIS LIMITATION SHALL ALSO APPLY WITH RESPECT TO DAMAGES INCURRED BY REASON OF OTHER SERVICES OR GOODS RECEIVED THROUGH OR ADVERTISED BY PELCO, AS WELL AS BY REASON OF ANY ADVICE RECEIVED THROUGH OR ADVERTISED BY PELCO. THIS LIMITATION SHALL ALSO APPLY, WITHOUT LIMITATION, TO THE COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, OR LOST DATA. SUCH LIMITATION SHALL FURTHER APPLY WITH RESPECT TO THE PERFORMANCE OR NON-PERFORMANCE OF THE API & SDK. SUCH LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY AND TO THE FULLEST EXTENT PERMITTED BY LAW.

SOME STATES OR OTHER JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS AND EXCLUSIONS MAY NOT APPLY TO YOU.

12. INDEMIFICATION.

YOU agree to indemnify, hold harmless, and defend PELCO from and against any claims or lawsuits, including attorneys' fees and costs, that arise out of or in connection with any software, software driver, code or product which YOU integrate with the Product or interfaces with the API & SDK Software, as well as with respect to any violation of Section 9 of this License.

13. CONFIDENTIALITY

YOU agree that YOU, and each of YOUR employees and representatives, shall maintain the confidentiality of the API & SDK Document and the API & SDK Software, and all accompanying materials.

14. SEVERABILITY.

If any provision of this License is found to be illegal, invalid, or unenforceable, that provision shall not affect the validity or enforceability of this License, and this License shall then be construed in all respects as if such invalid or unenforceable provision were omitted.

15. CONTROLLING LAW.

This License shall be governed by, and construed in accordance with, the laws of the State of California (without giving effect to principles of conflicts of laws). The venue for any legal proceedings arising from or connected with this Agreement shall be exclusively in Clovis, California, and each Party hereby waives the right to challenge venue based upon forum non conveniens or otherwise. The Parties expressly consent to the jurisdiction of the state and/or Federal courts in and/or for Clovis, California. This License shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.

16. GENERAL

This License constitutes the entire agreement between the parties with respect to the use of the API & SDK Document and the API & SDK Software licensed hereunder and supersedes all prior or contemporaneous understandings regarding such subject matter. No amendment to or modification of this License will be binding unless in writing and signed by an authorized representative of PELCO. Any translation of this License is done for local requirements and in the event of a dispute between the English and any non-English versions, the English version of this License shall govern. In any action to enforce any rights set forth in this License, the prevailing party shall be entitled to recover their attorneys' fees and costs, at all levels of trial and appeal. The failure or delay of either party to exercise any of its rights under this License shall not be deemed a waiver of those rights or the breach thereof.

QUESTIONS

Questions regarding the API & SDKDocument, the API & SDK Software, or this License and their implementation or use should be directed to:

PELCO
3500 PELCO Way
Clovis, California USA 93612-5699

(559) 292-1018 - Voice
(559) 292-1981 - FAX

Contact Information

WORLDWIDE HEADQUARTERS

3500 Pelco Way

Clovis, Ca 93612-5699

(800) 289-9100 USA & Canada

(800) 289-9150 USA & Canada Fax

(559) 292-1981 International

(559) 348-1120 International Fax

SALES

United States & Canada Sales | sales@pelco.com

International Sales | intl@pelco.com

TECHNICAL SUPPORT

Large Matrix Support | matrixsupport@pelco.com

Digital Support | digitalsupport@pelco.com

Questions related to PelcoNet™ and Pelco Digital Video Recorders

General Support | techsupport@pelco.com

API Support | apisupport@pelco.com

International Technical Support [intlTechSupport@pelco.com

MISCELLANEOUS

Literature Requests | lit@pelco.com

Press Contact | hcarder@pelco.com

Pelco Webmaster | webmaster@pelco.com